Technische Universität Berlin

FACHBEREICH 3
MATHEMATIK

LINEAR PRESELECTIVE POLICIES FOR STOCHASTIC PROJECT SCHEDULING

by

ROLF H. MÖHRING FREDERIK STORK

No. 612/1998(revised:August2000)

Linear Preselective Policies for Stochastic Project Scheduling

Rolf H. Möhring* Frederik Stork* ‡

June 30, 1998, revised August, 2000

Abstract

In the context of stochastic resource-constrained project scheduling we introduce a novel class of scheduling policies, the *linear preselective policies*. They combine the benefits of *preselective policies* and *priority policies*; two classes that are well known from both deterministic and stochastic scheduling. We study several properties of this new class of policies which indicate its usefulness for computational purposes. Based on a new representation of preselective policies as AND/OR *precedence constraints* we derive efficient algorithms for computing earliest job start times and state a necessary and sufficient dominance criterion for preselective policies.

A computational experiment based on 480 instances empirically validates the theoretical findings.

Keywords: Resource-constrained project scheduling – random processing times – scheduling policies – AND/OR precedence constraints.

1 Introduction

In resource-constrained project scheduling a set of *activities* or *jobs* has to be scheduled subject to both precedence and resource constraints in order to minimize some objective function. A recent survey on different settings and solution methods for this notoriously hard optimization problem has been contributed by Brucker, Drexl, Möhring, Neumann, and Pesch (1999). We consider here the case that job processing times are *random* according to a known probability distribution. This models the fact that real-world scheduling problems are quite often subject to uncertain events such as weather conditions, availability of employees, or obstruction of resource usage.

The combination of random job processing times and resource constraints leads into the area of stochastic dynamic programming. Scheduling is done by *policies* or *strategies*. A *policy* may be seen as an on-line rule that decides which jobs to start at certain decision times t, based on the knowledge of the given distributions and the

*Technische Universität Berlin, Department of Mathematics, Sekr. MA 6–1, Straße des 17. Juni 136, D-10623 Berlin, Germany, {moehring,stork}@math.tu–berlin.de, http://www.math.tu–berlin.de/coga.

‡The author is supported by the Deutsche Forschungsgemeinschaft (DFG) under grant Mo 446/3-3

observed past up to t. The best-known class of such scheduling policies is probably the class of *priority policies*. They order all jobs according to a *priority list* and, at every decision time t, start as many jobs as possible in the order of that list. While priority policies are easy to define and implement, they have several well known drawbacks. For example, there are instances (even with deterministic processing times) in which no priority policy yields an optimal schedule. Moreover, the change of job processing times may lead to so-called Graham anomalies such as an increasing project duration due to decreasing job processing times, see (Graham 1966). Thus, job start times resulting from priority policies are neither monotone nor continuous (when thinking of a policy as a function of the job processing times). The class of so-called *preselective policies* has been introduced by Igelmund and Radermacher (1983b). Roughly speaking, such policies define for each possible resource conflict a *preselected* job which is postponed if the corresponding resource conflict appears within the execution of the project. In contrast to priority policies, preselective policies do not show the undesired Graham anomalies and thus allow a safer planning. However, computational experience with this class by Igelmund and Radermacher (1983a) and Stork (1998) shows that calculating an optimal such policy requires excessive computation times.

In this paper, we combine the simplicity of priority policies with the structural attractiveness of preselective policies by defining the preselective jobs according to *priority lists*. The thus defined *linear preselective policies* inherit all the favorable properties of preselective policies but are significantly better computationally tractable. Based on a new representation of preselective policies as AND/OR *precedence constraints* we derive efficient algorithms for computing earliest job start times and state a necessary and sufficient dominance criterion for preselective policies. A computational experiment based on 480 randomly generated instances empirically validates the theoretical benefits of linear preselective policies. As a side result we show that the class of linear preselective policies properly dominates the class of *job-based priority rules* which is well known from deterministic scheduling, cf., e. g., (Patterson, Słowiński, Talbot, and Węglarz 1989).

The paper is organized as follows. In the next section we specify the considered scheduling model. We then formally define the classes of priority policies, job-based priority policies, and preselective policies in Section 3. Sections 4 and 5 are concerned with the definition of linear preselective policies and the discussion of resulting theoretical and computational benefits. In Section 6 we then compare the class of linear preselective policies to the classes introduced in Section 3.

2 Model and Notation

Let $V = \{1, \dots, n\}$ be a set of non-preemptable jobs. *Precedence constraints* are given by a set E_0 of ordered pairs (i, j) of jobs with the meaning that job j cannot be started before job i has been completed. These constraints define a partial order $G_0 = (V, E_0)$ on V. In addition, a finite set $\mathcal{R}$ of different, renewable resources is required to complete the project. A constant amount of R_k units of each resource $k \in \mathcal{R}$ is available throughout the project and each job j consumes r_{jk} units of resource $k \in \mathcal{R}$ while in process. At each point in time and for each k the sum of resource consumption of the jobs being performed must not exceed R_k. Such *resource constraints* can equivalently be represented by a system $\mathcal{F} \subseteq 2^V$ of so-called *forbidden sets*, i. e., inclusion-minimal sets F of pairwise not related jobs that cannot be scheduled simultaneously because they share some common limited resource. Throughout the paper we always consider this representation of resource constraints. Note that for particular settings of resource consumptions and supplies, the number of forbidden sets may grow exponentially in the number of jobs.

Processing times of jobs are not known in advance, but are instead given by a random vector $\boldsymbol{p} = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n)$ where $\boldsymbol{p}_i$ denotes the random processing time of job i. A particular sample of $\boldsymbol{p}$ is denoted by $p = (p_1, \ldots, p_n) \in \mathbb{R}^n_+$. The objective is to find a scheduling policy that minimizes a given regular cost function κ in expectation.

Assuming a short encoding of the job distributions, this variant of resource-constrained project scheduling is clearly strongly NP-hard, since it contains the NP-hard deterministic case. Following the classification scheme of Brucker, Drexl, Möhring, Neumann, and Pesch (1999), this problem is termed $PS|prec, p_j = sto|\kappa$.

In order to illustrate the presentation we use the following example (which has previously been used by Igelmund and Radermacher (1983b)).

Example 2.1. *(Igelmund and Radermacher 1983b) Let $G_0 = (V, E_0)$ be given by $V = \{1, 2, 3, 4, 5\}$ and $E_0 = \{(1, 4), (3, 5)\}$ and let the sets $F_1 = \{1, 5\}, F_2 = \{2, 3, 4\}$, and $F_3 = \{2, 4, 5\}$ be forbidden.*

3 Scheduling Policies

General scheduling policies. Due to the on-line character of stochastic scheduling problems with resource constraints, scheduling is done by *policies* or *strategies*, see (Möhring, Radermacher, and Weiss 1984; Möhring, Radermacher, and Weiss 1985) for a comprehensive characterization. We will here consider only *elementary* policies in their terms. Such a policy Π has the following dynamic interpretation. It chooses actions at decision points. *Decision points* are $t = 0$ (project start) and job completions. An action at time t consists of starting a *feasible set* of jobs at t, where feasible means that both precedence and resource constraints are respected. The decision may of course only exploit information that has become available until the current time t (plus the a-priori knowledge about the distribution of job processing times). Finally, when every job has been scheduled, we have a sample p of processing times and Π has constructed a *schedule* $S(p)$, i. e., a vector of job start times. Thus, every policy Π may be interpreted as a function $\Pi : p \to S(p)$ that maps given samples of job processing times to feasible schedules. We denote the start time of a job $j \in V$ obtained from a given policy Π and a sample p by $S^\Pi_j(p)$ and its completion time by $C^\Pi_j(p) := S^\Pi_j(p) + p_j$. The corresponding random variables are denoted by $\boldsymbol{S}^\Pi_j$ and $\boldsymbol{C}^\Pi_j$. If no misinterpretation is possible we omit the policy superscript Π.

Evaluation of schedules is done by the given performance measure κ. For a given sample p and associated completion times $C^\Pi(p)$ of jobs, $\kappa(C^\Pi(p))$ denotes the cost of the schedule resulting from p, and $E[\kappa(\boldsymbol{C}^\Pi)]$ denotes the expected cost under policy Π. The objective is to minimize κ *in expectation* (e. g. the expected project makespan) over a class of policies. We therefore define a stochastic scheduling policy Π^* to be optimal with respect to a certain class τ of policies if Π^* minimizes the expected project costs within τ: $E[\kappa(\boldsymbol{C}^{\Pi^*})] = \inf\{E[\kappa(\boldsymbol{C}^{\Pi})] : \Pi \in \tau\}$. We denote the cost of an optimum policy of class τ by ρ^τ.

Priority-Based Policies. We now outline the classes of *priority policies* and *job-based priority policies*. The policies of both classes have the same representation, namely a linear ordering L of the jobs. For any decision point t they start as many jobs as possible out of a given set $A(t)$ of *available jobs*, preferring those ones with a higher priority (which is defined by L). The only difference between both classes of policies is the definition of the sets $A(t)$ of available jobs. For the class of priority policies, $A(t)$ is defined as the set of all unscheduled jobs j whose predecessors have been completed by t. For job-based priority policies, $A(t)$ must additionally fulfill the following property. If $j \in A(t)$, then all jobs i with a higher priority (i. e., $i \prec_L j$) must either have been already scheduled ($S_i(p) < t$) or must also be contained in $A(t)$. Note that this condition makes only sense if L is a linear extension of the underlying partial order of precedence constraints.

Like traditional priority policies, job-based priority policies are also well known from deterministic scheduling. In the order of L, they start every job as early as possible with the side constraint that $S_i(p) \le S_j(p)$ if $i \prec_L j$. This "job-based" view instead of the "resource-based" view for priority policies is the reason for their name. Applications in deterministic scheduling occur in connection with approximation algorithms (Hall, Schulz, Shmoys, and Wein 1997) and *precedence-tree* enumeration schemes (Patterson, Słowiński, Talbot, and Węglarz 1989).

Preselective Policies. We next define the class of preselective policies. For a system $\mathcal{F} = \{F_1, \ldots, F_f\}$ of forbidden sets, a *selection* is a sequence $s = (s_1, \ldots, s_f)$ such that $s_\ell \in F_\ell$ for all $\ell \in \{1, \ldots, f\}$. A job $j \in F$ is called *waiting job* for the forbidden set F, if $\min_{i \in F \setminus \{j\}}\{S_i(p) + p_i\} \le S_j(p)$ is guaranteed for all $p \in \mathbb{R}^n_+$, i. e., the start of j is postponed until at least one job $i \in F \setminus \{j\}$ has been completed.

Definition 3.1. *A scheduling policy* Π *is called* preselective *with selection* $s = (s_1, \ldots, s_f)$*, if, for all* $\ell \in \{1, \ldots, f\}$*,* s_ℓ *is a waiting job for* F_ℓ*.*

Equivalently to selections we define *partial selections* $s^\ell := (s_1, \ldots, s_\ell)$, $\ell \in \{1, \ldots, f\}$, which can be seen as selections for the reduced system $\mathcal{F}^\ell := \{F_1, \ldots, F_\ell\}$ of forbidden sets. We therefore use the same notation as for selections.

Igelmund and Radermacher (1983b) have given the following combinatorial characterization of preselective policies: a preselective policy with selection s can be represented by a collection of partially ordered sets $G = (V, E)$ each of which extends the given partial order G_0 of precedence constraints and "respects" the selection s. That is, $E_0 \subseteq E$ and for each forbidden set F with preselected job $j \in F$, E contains an ordered pair (i, j) with $i \in F \setminus \{j\}$. We call such an extension of G_0 a *realization* of G_0 and s.

However, it is easier to make use of an alternative representation, so-called *waiting conditions*. Such conditions are given by pairs (X, j), $X \subset V$, $j \in V \setminus X$, where job

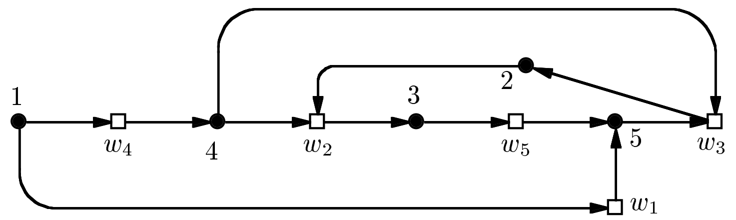

Figure 1: The digraph resulting from Example 2.1 and selection $s = (5, 3, 2)$. Circular nodes correspond to jobs while square nodes represent waiting conditions. Nodes w_1, w_2, and w_3 are induced by forbidden sets and the respective preselected job. Precedence constraints are represented by w_4 and w_5.

j cannot be started before at least one job $i \in X$ has been completed. Each restriction induced by a forbidden set F and its preselected job j can be represented by the waiting condition $(F \setminus \{j\}, j)$. Moreover, each given precedence constraint $(i, j) \in E_0$ can obviously be represented by $(\{i\}, j)$. Thus, instead of considering precedence constraints, forbidden sets and selections, it suffices to consider a set $\mathcal{W}$ of waiting conditions.

Note that a set of waiting conditions naturally induces a digraph D which has a node for each job and for each waiting condition. There is a directed arc from a node representing a job i to a node representing a waiting condition (X, j) if $i \in X$. Furthermore, each node representing a waiting condition (X, j) is connected to the node representing j. In the scheduling literature, the concept of waiting conditions is also known as AND/OR precedence constraints, see, e. g., (Gillies and Liu 1995), (Goldwasser and Motwani 1999), and (Möhring, Skutella, and Stork 2000b).

The set of waiting conditions which is induced by Example 2.1 with selection $s = (5, 3, 2)$ is $\mathcal{W} = \{w_1 = (\{1\}, 5), w_2 = (\{2, 4\}, 3), w_3 = (\{4, 5\}, 2), w_4 = (\{1\}, 4), w_5 = (\{3\}, 5)\}$. The associated digraph D representing $\mathcal{W}$ is depicted in Figure 1. Notice that, although D contains cycles, there is a possible ordering in which jobs can be executed ($1 \to 4 \to 3 \to 5 \to 2$). However, selections may be *infeasible*; which is the case if and only if there exists some non-empty set $V' \subseteq V$ such that for all $j \in V'$ there is a waiting condition (X, j) with $X \subseteq V'$ (Igelmund and Radermacher 1983a, Theorem 1.1). In this case no job from V' can be started. For Example 2.1 and $s = (1, 3, 2)$ the set V fulfills this property. The selection s is therefore infeasible and does not induce a preselective policy. An linear time algorithm (in the encoding of $\mathcal{W}$) which detects infeasiblity has been proposed by Möhring, Skutella, and Stork (2000b).

For a given feasible selection s and resulting system of waiting conditions $\mathcal{W}$ we can construct a feasible schedule for each possible sample $p \in \mathbb{R}^n_+$ by setting

$$S_j(p) = \max_{(X,j) \in \mathcal{W}_j} \{\min_{i \in X} \{S_i(p) + p_i\}\} \tag{1}$$

We hereby assume that the maximum over an empty set is 0. $\mathcal{W}_j \subseteq \mathcal{W}$ denotes the set of waiting conditions (X, j). Notice that $S_j(p)$ is in fact the earliest possible start of j with respect to $\mathcal{W}$ and p. In Section 5 below we discuss the algorithmic treatment of this recursive formula. When interpreting a preselective policy as a function $p \to S(p)$ defined by (1), Radermacher (1984) has shown the following properties.

Theorem 3.2. *(Radermacher 1984) A policy Π is preselective if and only if Π is continuous. Furthermore, Π is preselective if and only if Π is monotonically increasing.*

It follows that unfavorable effects such as the Graham anomalies (Graham 1966) will not occur when planning with a preselective policy. Another consequence is the fact that the class of preselective policies is *stable*. This means that a slight change of the distribution of job processing times only has a slight influence on the optimum value of this class for continuous cost functions κ, see (Radermacher 1984) and (Möhring, Radermacher, and Weiss 1984, Theorem 3.1.2).

4 Linear Preselective Policies

We are now ready to define *linear preselective policies* which combine the list-oriented features of priority policies with the selection-oriented character of preselective policies. The idea is to define the selection via a priority list. If L is this list, then the preselected job of a forbidden set F is the one with lowest priority in L, i. e., the last element of F in L. We denote this job by $j = \max(L|F)$.

Definition 4.1. *Let G_0 be a partially ordered set of jobs with an associated system $\mathcal{F}$ of forbidden sets $\{F_1, \ldots, F_f\}$ and let Π be a preselective policy with selection $s = (s_1, \ldots, s_f)$. Π is called* linear preselective *if there exists a linear extension L of G_0 such that for each forbidden set F_ℓ, $\ell \in \{1, \ldots, f\}$, the associated preselected job s_ℓ is the job of F_ℓ with lowest priority with respect to L, i. e., $s_\ell = \max(L|F_\ell)$.*

Theorem 4.2. *Let G_0 be a partially ordered set of jobs with an associated system $\mathcal{F}$ of forbidden sets $\{F_1, \ldots, F_f\}$.*

(1) Every linear extension of G_0 induces a linear preselective policy.

(2) A preselective policy Π with an associated selection $s = (s_1, \ldots, s_f)$ is linear preselective if and only if the digraph D induced by the set $\mathcal{W}$ of waiting conditions corresponding to s is acyclic.

Proof. We only give a proof of claim (2), (1) follows immediately from the definition of linear preselective policies. Let the list L correspond to a linear preselective policy. We extend the list to a list of all nodes of D by inserting the node representing a forbidden set F directly before the last job of F. Then L defines a topological sort of the nodes of D and thus D is acyclic. Conversely, let D be acyclic and let L be an ordering of the jobs obtained from a topological sort of D by ignoring the nodes that correspond to the forbidden sets. It follows from the definition of D that, for all waiting conditions (X, j) represented by D, j must be the largest job in $L|X$. Consequently, the preselected job j of each forbidden set F is maximal in $L|F$. Furthermore, since each precedence constraint (i, j) is modeled by a waiting condition $(\{i\}, j)$, L is an extension of G_0. Therefore, the preselective policy corresponding to D is linear preselective. $\square$

As a direct consequence we obtain that linear preselective policies allow a very flexible execution of the project: suppose that for each F_ℓ we have to fix a job $i_\ell \in F_\ell \setminus \{s_\ell\}$ for which s_ℓ should wait. The policy induced by the selection s is linear preselective if and only if any combination of such choices is feasible.

Corollary 4.3. *Let $\bar{E}$ denote a set of ordered pairs $\{(i_\ell, s_\ell) : \ell \in \{1, \ldots, f\}\}$ where each i_ℓ is arbitrarily chosen from $i_\ell \in F_\ell \setminus \{s_\ell\}$. A preselective policy Π is linear preselective if and only if every set $\bar{E}$ of such choices yields a realization (i. e., $E_0 \cup \bar{E}$ is acyclic).*

The corollary follows directly from (2) of Theorem 4.2.

5 Benefits of Linear Preselective Policies

5.1 Computing Earliest Job Starting Times

Hagstrom (1988) has shown that, even when disregarding all resource constraints, the determination of the expected project makespan is #P-complete if each job processing time distribution has two (discrete) values. Although for complex distributions the problem has a longer encoding which could possibly admit a polynomial algorithm, such an efficient algorithm seems very unlikely to exist, since Hagstrom has also shown that — unless P=NP — the problem cannot be solved in time polynomial in the number of possible values of the makespan (assuming discrete distributions). For these reasons one usually approximates the expected cost of a given policy by applying *simulation* to the given distributions of job processing times. We then obtain a set P of samples for $\boldsymbol{p}$, that can be used to approximately calculate $E[\kappa(\boldsymbol{C}^\Pi)]$. This is done by computing the average cost of the schedules resulting from these samples, i. e., $E[\kappa(\boldsymbol{C}^\Pi)] \approx \frac{1}{|P|}\sum_{p \in P} \kappa(C^\Pi(p))$.

For a given preselective policy and a sample of the job processing times, we compute earliest job starting times according to the recursive formula (1). Igelmund and Radermacher (1983a) proposed an algorithm with a running time $\mathrm{O}(mn^2)$, where $m := E_0 + \sum_{F \in \mathcal{F}} |F|$. However, for AND/OR precedence constraints there are better algorithms available. When applying the most efficient algorithm for this model (Möhring, Skutella, and Stork 2000b) the complexity reduces to $\mathrm{O}(n + m + f \log f)$ where $f = |\mathcal{F}|$. Note that, even with $f = \mathrm{O}(2^n)$, the algorithm has a lower worst case running time than the one proposed by Igelmund and Radermacher (1983a).

The algorithm is a modification of Dijkstra's shortest path algorithm. It operates on the digraph D defined by the set of waiting conditions resulting from G_0 and the given selection. Start times S are computed for all nodes of D where $S_j := \max_{w \in \mathcal{W}_j} S_w$ for nodes j corresponding to jobs and $S_w := \min_{i \in X}\{S_i + p_i\}$ for nodes w that are associated with a waiting condition (X, j). The nodes w corresponding to waiting conditions are maintained in a heap where the priority key is their current 'start time' S_w (initially $S_w = \infty$). The start times of all jobs that are not a waiting job of any waiting condition are initialized to 0. We then proceed over time by always choosing a node corresponding to a waiting condition (X, j) with minimum start time from the heap and plan that node at its current start time (i. e., we permanently fix the start time of that node). If all other "nodes" $(X', j) \in \mathcal{W}_j$ preceding j have already been planned we also plan j at the current time. Subsequently the start times of all waiting conditions $w = (X, i)$ with $j \in X$ are updated to $S_w := \min(S_w, S_j + d_{jw})$. Finally, if the heap is empty the algorithm outputs S. For further algorithmic details we refer to (Möhring, Skutella, and Stork 2000b).

For linear preselective policies, we can exploit the fact that the selection s of such a policy can be represented by a linear extension L of the underlying partial order of precedence constraints. By calculating the start times in the order of L, we can guarantee that for each sample $p \in \mathbb{R}^n_+$ of job processing times and each forbidden set F with preselected job j, the start times of all jobs $i \in F \setminus \{j\}$ have been computed before S_j is computed. In particular, no heap is required and the running time for linear preselective policies thus reduces to $\mathrm{O}(n + m)$.

5.2 Domination

We next discuss domination properties of preselective policies. We call a policy Π *dominated* if there exists another policy $\Pi' \neq \Pi$ such that $S^{\Pi}(p) \geq S^{\Pi'}(p)$ for all samples $p \in \mathbb{R}^n_+$ (Π and Π' are assumed to belong to the same class of policies). We derive a necessary and sufficient dominance criterion for preselective policies and show that many dominated preselective policies are not linear preselective.

The domination criterion is based on the property that preselected jobs defined for some forbidden sets may additionally solve the conflict on other forbidden sets. Suppose that for a partial selection $s^{\ell-1} = (s_1, ..., s_{\ell-1})$ there exists some $j \in F_\ell$ such that each realization $G = (V, E)$ of G_0 and $s^{\ell-1}$ contains a pair $(i, j) \in E$ with $i \in F_\ell \setminus \{j\}$. Then the resource conflict due to F_ℓ never occurs and F_ℓ can be discarded from further considerations. We say that F_ℓ is *implicitly resolved* by j. The intuition of the dominance criterion is that every extension of the partial selection $s^{\ell-1}$ to a selection s where j is not chosen as the preselected job for F_ℓ is dominated.

Theorem 5.1. *A preselected policy Π with selection $s = (s_1, \ldots, s_f)$ is dominated if and only if the resource conflict due to some forbidden set F_ℓ ($\ell = \{1, \ldots, f\}$) is implicitly resolved by $j \in F_\ell \setminus \{s_\ell\}$.*

In order to show Theorem 5.1 we again use the concept of waiting conditions. Similarly to preselective policies and selections we say that a set of waiting conditions $\mathcal{W}'$ dominates $\mathcal{W}$ if, for all samples $p \in \mathbb{R}^n_+$, the corresponding earliest start schedules $S(p)$ and $S'(p)$ obtained from (1) fulfill $S'(p) \leq S(p)$. Moreover, we also consider *realizations* of a set $\mathcal{W}$ of waiting conditions. For each $(X, j) \in \mathcal{W}$ an ordered pair (i, j) with $i \in X$ is chosen such that the resulting set of pairs does not contain a cycle. The transitive closure of these pairs is called a realization of $\mathcal{W}$. If for some $(U, j) \notin \mathcal{W}$ ($j \notin U$) there exists a pair (i, j) with $i \in U$ for each realization of $\mathcal{W}$ we say that (U, j) is *implied* by $\mathcal{W}$. Note that this is particularly the case if $U \supseteq X$ for some $(X, j) \in \mathcal{W}$. (U, j) may be seen as an additional waiting condition that is implied by $\mathcal{W}$.

For the proof of Theorem 5.1 we require the following observations.

Lemma 5.2. *Let $\mathcal{W}$ and $\mathcal{W}'$ be feasible sets of waiting conditions. $\mathcal{W}$ is dominated by $\mathcal{W}'$ if and only if each $(X, j) \in \mathcal{W}'$ is implied by $\mathcal{W}$.*

Proof. For an arbitrary sample $p \in \mathbb{R}_+^n$ let $S(p)$ and $S'(p)$ denote the earliest start schedules obtained by applying (1) to $\mathcal{W}$ and $\mathcal{W}'$, respectively. If each $(X, j) \in \mathcal{W}'$ is implied by $\mathcal{W}$ then $S(p)$ must respect at least as many constraints as $S'(p)$ and therefore $S'(p) \leq S(p)$.

Let $(X, j) \in \mathcal{W}'$ and suppose that (X, j) is not implied by $\mathcal{W}$. Set $p_i := 1$ if $i \in X$ and $p_i := 0$, otherwise and recall that each (X', j) with $X' \subseteq X$ implies (X, j). We obtain $S'_j = 1$ while $S_j = 0$, which is a contradiction to the fact that $\mathcal{W}$ is dominated by $\mathcal{W}'$. $\square$

Lemma 5.3. *If $(U \cup \{h\}, j)$ is implied by a given system $\mathcal{W}$ of waiting conditions with $(U \cup \{j\}, h) \in \mathcal{W}$, then $(U \cup \{h\}, j)$ is also implied by the reduced system $\mathcal{W}' := \mathcal{W} \setminus \{(U \cup \{j\}, h)\}$.*

Proof. Suppose that there exists a realization $G = (V, E)$ of $\mathcal{W}'$ containing no pair $(i, j) \in E$ with $i \in U \cup \{h\}$. If G contains no pair (i, h) with $i \in U \cup \{j\}$ we add an arbitrary such pair (i, h) with $(h, i) \notin E$ to E (plus resulting transitive pairs). Such a job i always exists because $(h, j) \notin E$ and thus j fulfills the required properties. We obtain a realization of $\mathcal{W}$. In this realization there exists no pair (i, j) with $i \in U \cup \{h\}$ which is a contradiction to the fact that $(U \cup \{h\}, j)$ is implied by $\mathcal{W}$. $\square$

Proof of Theorem 5.1 Suppose the resource conflict due to some forbidden set F_ℓ is implicitly resolved by $j \in F_\ell \setminus \{s_\ell\}$. Define the selection s' by $s'_r := s_r$ for $r \in \{1, \ldots, f\} \setminus \{\ell\}$ and $s'_\ell := j$. Let $\mathcal{W}$ and $\mathcal{W}'$ denote the sets of waiting conditions induced by s and s'. By assumption, $(F_\ell \setminus \{j\}, j)$ is implied by $\mathcal{W}$. By Lemma 5.3 this is also the case for $\mathcal{W} \setminus \{(F_\ell \setminus \{s_\ell\}, s_\ell)\}$. Since $\mathcal{W}' = \mathcal{W} \setminus \{(F_\ell \setminus \{s_\ell\}, s_\ell)\} \cup \{(F_\ell \setminus \{j\}, j)\}$ it follows that each $(X, j) \in \mathcal{W}'$ is implied by $\mathcal{W}$. Therefore, with Lemma 5.2, $\mathcal{W}$ is dominated by $\mathcal{W}'$ and it directly follows that Π is dominated by the policy Π' induced by s'.

For the converse claim consider a policy Π' with selection s' that dominates Π. For the induced sets of waiting conditions it follows by definition of dominance that $\mathcal{W}'$ dominates $\mathcal{W}$. By Lemma 5.2, all $(F_\ell \setminus \{s'_\ell\}, s'_\ell) \in \mathcal{W}'$ ($\ell \in \{1, \ldots, f\}$) are implied by $\mathcal{W}$. It thus follows for Π that for all $\ell \in \{1, \ldots, f\}$ with $s_\ell \neq s'_\ell$ the resource conflict due to F_ℓ is implicitly resolved by s'_ℓ .

Möhring, Skutella, and Stork (2000b) present an algorithm that, for a given set $\mathcal{W}$ of waiting conditions and a (forbidden) set $F \subset V$ of jobs, computes all waiting jobs for F in linear time (in the encoding of V and $\mathcal{W}$). By Theorem 5.1, when applying the algorithm for each forbidden set, it can be decided in $O(f \cdot (n + m))$ time whether a given preselective policy is dominated or not.

We now show that quite a few dominated preselective policies are not linear preselective and thus, when dealing with linear preselective policies all these dominated policies are discarded beforehand. Consider a partial selection $s^{\ell-1} = (s_1, \ldots, s_{\ell-1})$ inducing a set $\mathcal{W}$ of waiting conditions which imply the waiting condition $(F_\ell \setminus \{j\}, j)$. Defining j as the preselected job for F_ℓ thus creates no further restrictions. Contrarily, any $i \in F_\ell \setminus \{j\}$ which is chosen to be preselected for F_ℓ causes an additional waiting condition $(F_\ell \setminus \{i\}, i)$. Thus, any such preselective policy is dominated.

Corollary 5.4. *Let $s^{\ell-1}$, $\mathcal{W}$, and $j \in F_\ell$ be as above. Denote by $U \subseteq F_\ell \setminus \{j\}$ the set of jobs i such that there exists a directed path from i to some other job $h \in F_\ell \setminus \{i\}$ in the digraph D resulting from $\mathcal{W}$. Then all preselective policies that are induced by extending $s^{\ell-1}$ to a selection s with $s_\ell \in U$ are dominated and not linear preselective.*

Proof. It follows from Theorem 5.1 that Π is dominated if $s_\ell \neq j$. This particularly holds for all $s_\ell \in U$. However, choosing s_ℓ from U would induce a cycle in D and thus, by Theorem 4.2, Π is not linear preselective. □

In Example 2.1, there exist $2 \cdot 3 \cdot 3 = 18$ selections (including one infeasible selection). As it can easily be verified, 12 of them are dominated by other selections. Among these dominated selections, only one corresponds to a linear preselective policy while all non-dominated policies are linear preselective. Suppose that we have preselected job 3 with respect to the forbidden set $\{2, 3, 4\}$. Then, job 3 and thus also job 5 is waiting for either job 2 or 4. In the preselective case we could then choose job 2 to be the preselected job for the forbidden set $\{2, 4, 5\}$. This leads to a dominated policy because $\{2, 4, 5\}$ is implicitly resolved by job 5. Contrarily, in the linear preselective case, in all orderings L of jobs which allow job 3 to be preselected with respect to $\{2, 3, 4\}$ we have $2 \prec_L 5$ and $4 \prec_L 5$ and thus, job 5 is the only possible choice of a preselected job for $\{2, 4, 5\}$.

We finally describe another characteristic of linear preselective policies which is related to Corollary 5.4. Let $s^{\ell-1}$, F_ℓ, and $j \in F_\ell$ be as above, i. e., F_ℓ is implicitly resolved by the waiting condition $(F_\ell \setminus \{j\}, j)$ with respect to $s^{\ell-1}$. Then, a preselective policy which is induced by extending $s^{\ell-1}$ to a selection s with $s_\ell = j$ is not necessarily linear preselective. In this case, every linear preselective policy which is induced by extending $s^{\ell-1}$ to a selection s includes a superfluous waiting condition $(F_\ell \setminus \{s_\ell\}, s_\ell)$. However, notice that such waiting conditions can be identified (and removed) by the above mentioned algorithm developed by Möhring, Skutella, and Stork (2000b). We make use of such superfluous waiting conditions in Example 6.1 below where we relate the optimum values of preselective and linear preselective policies.

5.3 Memory Requirements

Another important computational benefit of linear preselective policies is their compact representation. Obviously, in contrast to preselective policies, for a given set $\mathcal{F}$ of forbidden sets, every linear preselective policy can be stored within $\mathrm{O}(n)$ space. We simply use the linear ordering L of the jobs to represent the policy. Then, for a given forbidden set F, we may obtain the corresponding preselected job by identifying the largest job of F with respect to L. This is extremely helpful, particularly in the context of branch-and-bound approaches where typically a large number of policies has to be maintained for the same set $\mathcal{F}$.

5.4 Summary and Computational Results

To recapitulate the above mentioned benefits, we obtain the following proposition.

Algorithm	100 sec	200 sec	500 sec	1000 sec
LIN	414	437	453	461
PRS	284	301	318	325
LIN (dom)	450	460	466	467
PRS (dom)	367	383	393	406

Table 1: Results have been obtained on a Sun Ultra 1 with 143 MHz clock pulse operating under Solaris 2.7. The code was written in C++ and is compiled with the GNU g++ compiler version 2.91.66 using the -O3 optimization option. The test set contains 480 instances each of which consists of 20 jobs. They were created randomly by the widely used instance generator ProGen developed by Kolisch and Sprecher (1996). The figures are based on 200 samples of job processing times per instance. Each entry displays the number of instances that could be solved optimally by the variant of the program (rows) and within the given time limit (columns).

Proposition 5.5. *When planning with linear preselective policies instead of preselective policies we obtain the following benefits.*

(1) The computation of expected project costs can be performed more efficiently, because no heap (which costs $f \log f$ for each considered sample of processing times) is required within the computation.

(2) In the sense of Corollary 5.4, many dominated preselective policies are discarded from consideration.

(3) If a large number b of policies must be maintained, we save memory of size $b \cdot (f - n)$.

In fact, within a branch-and-bound framework, the properties of linear preselective policies mentioned in Proposition 5.5 result in a considerable speedup when compared with the class of preselective policies. In Table 1 we report on computational results of both classes of policies. A comparison of Rows 1 and 2 indicates that substantially more instances can be solved optimally by the linear preselective approach within given time limits. Rows 3 and 4 display results if a dominance rule based on Theorem 5.1 is enabled. Again, many more instances can be solved to optimality in the linear preselective case. Comparing the sizes of the search trees we observe that roughly 25% more nodes have to be considered in the preselective case (averaged over all instances solved optimally by both approaches). This fact empirically validates (2) of Proposition 5.5. With respect to (1) we considered the total number of nodes that could be investigated within a given time-limit. Compared to the preselective approach we could evaluate roughly 25% more nodes in the linear preselective case.

A detailed study on the computation of optimum policies in stochastic resource-constrained project scheduling has been carried out by Stork (1998).

6 Optimum Expected Costs of Linear Preselective Policies

We now compare the class of linear preselective policies with other classes of policies with respect to their optimum expected project cost ρ. To simplify notation we introduce the following identifiers for the classes of policies introduced in Section 3. We denote the classes of preselective and linear preselective policies by PRS and LIN, respectively. The traditional (resource-based) priority policies are referred to as RBP and the class of job-based priority policies is denoted by JBP.

6.1 Preselective Policies

Since each linear preselective policy is also preselective, we have $\rho^{\mathrm{PRS}} \leq \rho^{\mathrm{LIN}}$. But, in fact, the computational experiments reported in Section 5.4 (and also those of Stork (1998)) show that the subclass of linear preselective policies yields excellent objective values within the class of preselective policies. In particular, within reasonable numerical precision, the optimum expected costs are identical for all instances that have been considered. However, there exist artificial instances for which $\rho^{\mathrm{PRS}} < \rho^{\mathrm{LIN}}$ is possible. We give two counterexamples. The first one is based on domination properties of preselective policies as discussed in Section 5.2.

Example 6.1. *Let $V = \{1, \ldots, 8\}$ and $E_0 = \{(2,6), (3,7), (4,8)\}$. Forbidden sets are given by $F_1 = \{1,6\}$, $F_2 = \{6,7\}$, $F_3 = \{5,7,8\}$, and $F_4 = \{1,7,8\}$. The job processing times are distributed as follows: $p_1 \in \{1,15\}$, $p_2 = 8$, $p_3 = 12$, $p_4 = 14$, $p_5 = 14$, $p_6 = 4$, $p_7 = 2$, $p_8 \in \{1,15\}$. Only the processing times of jobs 1 and 8 are random, each value appears with probability $\frac{1}{2}$, independently from each other.*

The unique optimal preselective policy is $s^{\mathrm{PRS}} = (6,7,8,7)$ while the best linear preselective policy is $s^{\mathrm{LIN}} = (6,7,8,8)$; we obtain $\rho^{\mathrm{PRS}} = 23.5 < 23.75 = \rho^{\mathrm{LIN}}$. The gap stems from the waiting condition $(\{1,7\},8)$ which has to be respected according to s^{LIN}. Notice that $(\{1,7\},8)$ is superfluous since F_4 is implicitly resolved due to the choices of preselected jobs for F_1 and F_2.

We next provide another simple counterexample in which job processing times are dependent.

Example 6.2. *Let $V = \{1,2,3,4\}, E_0 = \emptyset$, and $F_1 = \{1,2,3\}, F_2 = \{1,2,4\}$. Furthermore, the following four samples of processing times are possible, each with probability $\frac{1}{4}$. $p^1 = (2,2,1,5), p^2 = (2,2,5,1), p^3 = (6,2,1,5)$ and $p^4 = (2,6,5,1)$. The objective is to minimize the expected makespan.*

A straightforward case analysis shows that the non-linear preselective selection $s = (1,2)$ is the unique optimal solution in the class of preselective policies. We obtain $\rho^{\mathrm{PRS}} = 6 < 6.25 = \rho^{\mathrm{LIN}}$.

6.2 Priority-Based Policies

We now compare the class of linear preselective policies with the two classes of priority policies considered in Section 3. All these classes have in common that each policy can be represented by a linear ordering L of the jobs but they make use of distinct algorithms to transform a given sample of job processing times into a schedule.

It is well known and can be verified by simple instances that the class of preselective policies and the class of priority policies are incomparable, i. e., there exist instances with $\rho^{\text{PRS}} < \rho^{\text{RBP}}$ and vice versa. For linear preselective policies we obtain the same result, as can be shown by the same examples as for preselective policies.

We next compare linear preselective policies and job-based priority policies (JBP). Recall that, for job-based priority policies, a schedule for a given sample p is computed by the algorithm described in Section 3.

Theorem 6.3. *If a job-based priority policy Π and a linear preselective policy Π' are induced by the same linear ordering L of jobs, then Π' dominates Π.*

Proof. Consider a forbidden set F and let j be its preselected job with respect to Π', i. e., $j = \max(L|F)$. We show that j is a waiting job for F with respect to Π. For an arbitrary sample $p \in \mathbb{R}^n_+$ let $S(p)$ be the schedule obtained from Π. Since $S_i(p) \le S_j(p)$ for all $i \prec_L j$ this also holds for all $i \in F \setminus \{j\}$. Consequently, since the resource conflicts of all forbidden sets are solved in S, we have $S_j(p) \ge \min_{i \in F\setminus\{j\}}\{S_i(p) + p_i\}$, i. e., j is a waiting job for F. It follows that Π induces at least as many constraints as Π' and thus, Π' dominates Π. □

As a direct consequence of Theorem 6.3 we obtain that $\rho^{\text{LIN}} \le \rho^{\text{JBP}}$. This in turn implies that optimal schedules for deterministic problems can be computed via linear preselective policies.

Corollary 6.4. *Given a deterministic resource-constrained project scheduling problem with job processing times p, there exists a linear preselective policy Π such that $S^{\Pi}(p)$ is an optimal schedule.*

Proof. Since it is known for deterministic scheduling that an optimal schedule may be obtained by a job-based priority policy, the result follows from Theorem 6.3. □

Notice that AND/OR precedence constraints are also useful to express job-based priority policies. Let Π and Π' be as in Theorem 6.3 and consider the set $\mathcal{W}'$ of waiting conditions resulting from Π'. We iteratively construct a set $\mathcal{W} \supseteq \mathcal{W}'$ that represents Π. First, set $\mathcal{W} := \mathcal{W}'$. Then, for each j in the order of L add all (X, j) with $(X, i) \in \mathcal{W}$ to $\mathcal{W}$ (where i denotes the job which immediately precedes j in L). It is not hard to see that, for each p, the earliest start schedule obtained from $\mathcal{W}$ according to (1) and the schedule S resulting from Π coincide. Hence, we obtain the following result.

Theorem 6.5. *Let G_0 be a partially ordered set of jobs with an associated system $\mathcal{F}$ of forbidden sets. If Π is a job-based priority policy for G_0 and $\mathcal{F}$ then Π is linear preselective for G_0 and a larger system $\mathcal{F}' \supseteq \mathcal{F}$ of forbidden sets.*

As a direct consequence of Theorem 6.5, when interpreting a job-based priority policy Π as a function $p \to S^{\Pi}(p)$, Π is monotone and continuous and thus, by (Möhring, Radermacher, and Weiss 1984, Theorem 3.1.2), the class of job-based priority policies is stable.

An interesting behavior occurs for parallel machine scheduling $P|p_j = sto|\kappa$ without any temporal constraints (e. g., release dates or precedence constraints). In this model the algorithms of all considered list-based scheduling policies generate identical schedules, i. e., LIN = RBP = JBP.

7 Concluding Remarks

We have invented the class of linear preselective policies to solve stochastic resource-constrained project scheduling problems. This novel class of scheduling policies combines the benefits of the previously known classes of preselective policies and priority policies. We have shown that linear preselective policies have a compact representation and allow a more efficient computation of expected project costs when compared to preselective policies. Furthermore, we proposed a necessary and sufficient dominance criterion for preselective policies. Most of the results have been obtained by the idea of representing scheduling policies as AND/OR precedence constraints. This in total leads to a favorable basis for an approach to compute stable, high quality solutions for stochastic resource-constrained project scheduling problems at reasonable computational expense.

References

Brucker, P., A. Drexl, R. H. Möhring, K. Neumann, and E. Pesch (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research 112*, 3–41.

Gillies, D. W. and J. W.-S. Liu (1995). Scheduling tasks with AND/OR precedence constraints. *SIAM Journal on Computing 24*, 797–810.

Goldwasser, M. H. and R. Motwani (1999). Complexity measures for assembly sequences. *International Journal of Computational Geometry and Applications 9*, 371–418.

Graham, R. L. (1966). Bounds on multiprocessing timing anomalies. *Bell System Technical Journal 45*, 1563–1581.

Hagstrom, J. N. (1988). Computational complexity of PERT problems. *Networks 18*, 139–147.

Hall, L. A., A. S. Schulz, D. B. Shmoys, and J. Wein (1997). Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Mathematics of Operations Research 22*, 513–544.

Igelmund, G. and F. J. Radermacher (1983a). Algorithmic approaches to preselective strategies for stochastic scheduling problems. *Networks 13*, 29–48.

Igelmund, G. and F. J. Radermacher (1983b). Preselective strategies for the optimization of stochastic project networks under resource constraints. *Networks 13*, 1–28.

Kolisch, R. and A. Sprecher (1996). PSPLIB - a project scheduling problem library. *European Journal of Operational Research 96*, 205–216.

Möhring, R. H., F. J. Radermacher, and G. Weiss (1984). Stochastic scheduling problems I: General strategies. *ZOR – Zeitschrift für Operations Research 28*, 193–260.

Möhring, R. H., F. J. Radermacher, and G. Weiss (1985). Stochastic scheduling problems II: Set strategies. *ZOR – Zeitschrift für Operations Research 29*, 65–104.

Möhring, R. H., M. Skutella, and F. Stork (2000a). Forcing relations for AND/OR precedence constraints. In *Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'00)*, pp. 235–236.

Möhring, R. H., M. Skutella, and F. Stork (2000b). Scheduling with AND/OR precedence constraints. Technical Report 689/2000, Technische Universität Berlin, Department of Mathematics, Germany. A preliminary version of this paper (*Forcing Relations for AND/OR Precedence Constraints*) appeared in (Möhring, Skutella, and Stork 2000a).

Patterson, J. H., R. Słowiński, F. B. Talbot, and J. Węglarz (1989). An algorithm for a general class of precedence and resource constrained scheduling problems. In R. Słowiński and J. Węglarz (Eds.), *Advances in Project Scheduling*, pp. 3–28. Elsevier.

Radermacher, F. J. (1984). Optimale Strategien für stochastische Scheduling Probleme. Habilitationsschrift, Rheinisch-Westfälische Technische Hochschule Aachen.

Stork, F. (1998). A branch-and-bound algorithm for minimizing expected makespan in stochastic project networks with resource constraints. Technical Report 613/1998, Technische Universität Berlin, Department of Mathematics, Germany. Revised July 2000.

Reports from the group

"Combinatorial Optimization and Graph Algorithms"

of the Department of Mathematics, TU Berlin

693/2000 *Frederik Stork and Marc Uetz:* On the Representation of Resource Constraints in Project Scheduling

691/2000 *Martin Skutella and Marc Uetz:* Scheduling Precedence Constrained Jobs with Stochastic Processing Times on Parallel Machines

689/2000 *Rolf H. Möhring, Martin Skutella, and Frederik Stork:* Scheduling with AND/OR precedence constraints

685/2000 *Martin Skutella:* Approximating the single source unsplittable min-cost flow problem

684/2000 *Han Hoogeveen, Martin Skutella, and Gerhard J. Woeginger:* Preemptive scheduling with rejection

683/2000 *Martin Skutella:* Convex quadratic and semidefinite programming relaxations in Scheduling

682/2000 *Rolf H. Möhring and Marc Uetz:* Scheduling Scarce Resources in Chemical Engineering

681/2000 *Rolf H. Möhring:* Scheduling under Uncertainty: Optimizing Against a Randomizing Adversary

680/2000 *Rolf H. Möhring, Andreas S. Schulz, Frederik Stork, and Marc Uetz:* Solving Project Scheduling Problems by Minimum Cut Computations (Journal version for the previous Reports 620 and 661)

674/2000 *Esther M. Arkin, Michael A. Bender, Erik D. Demaine, Sándor P. Fekete, Joseph S. B. Mitchell, and Saurabh Sethia:* Optimal Covering Tours with Turn Costs

669/2000 *Michael Naatz:* A Note On a Question of C. D. Savage

667/2000 *Sándor P. Fekete and Henk Meijer:* On Geometric Maximum Weight Cliques

666/2000 *Sándor P. Fekete, Joseph S. B. Mitchell, and Karin Weinbrecht:* On the Continuous Weber and k-Median Problems

664/2000 *Rolf H. Möhring, Andreas S. Schulz, Frederik Stork, and Marc Uetz:* A Note on Scheduling Problems with Irregular Starting Time Costs

661/2000 *Frederik Stork and Marc Uetz:* Resource-Constrained Project Scheduling: From a Lagrangian Relaxation to Competitive Solutions

658/1999 *Olaf Jahn, Rolf H. Möhring, and Andreas S. Schulz:* Optimal Routing of Traffic Flows with Length Restrictions in Networks with Congestion

655/1999 *Michel X. Goemans and Martin Skutella:* Cooperative facility location games

654/1999 *Michel X. Goemans, Maurice Queyranne, Andreas S. Schulz, Martin Skutella, and Yaoguang Wang:* Single Machine Scheduling with Release Dates

653/1999 *Andreas S. Schulz and Martin Skutella:* Scheduling unrelated machines by randomized rounding

646/1999 *Rolf H. Möhring, Martin Skutella, and Frederik Stork:* Forcing Relations for AND/OR Precedence Constraints

640/1999 *Foto Afrati, Evripidis Bampis, Chandra Chekuri, David Karger, Claire Kenyon, Sanjeev Khanna, Ioannis Milis, Maurice Queyranne, Martin Skutella, Cliff Stein, and Maxim Sviridenko:* Approximation Schemes for Minimizing Average Weighted Completion Time with Release Dates

639/1999 *Andreas S. Schulz and Martin Skutella:* The Power of α-Points in Preemptive Single Machine Scheduling

634/1999 *Karsten Weihe, Ulrik Brandes, Annegret Liebers, Matthias Müller–Hannemann, Dorothea Wagner and Thomas Willhalm:* Empirical Design of Geometric Algorithms

633/1999 *Matthias Müller–Hannemann and Karsten Weihe:* On the Discrete Core of Quadrilateral Mesh Refinement

632/1999 *Matthias Müller–Hannemann:* Shelling Hexahedral Complexes for Mesh Generation in CAD

631/1999 *Matthias Müller–Hannemann and Alexander Schwartz:* Implementing Weighted b-Matching Algorithms: Insights from a Computational Study

629/1999 *Martin Skutella:* Convex Quadratic Programming Relaxations for Network Scheduling Problems

628/1999 *Martin Skutella and Gerhard J. Woeginger:* A PTAS for minimizing the total weighted completion time on identical parallel machines

627/1998 *Jens Gustedt:* Specifying Characteristics of Digital Filters with FilterPro

620/1998 *Rolf H. Möhring, Andreas S. Schulz, Frederik Stork, and Marc Uetz:* Resource Constrained Project Scheduling: Computing Lower Bounds by Solving Minimum Cut Problems

619/1998 *Rolf H. Möhring, Martin Oellrich, and Andreas S. Schulz:* Efficient Algorithms for the Minimum-Cost Embedding of Reliable Virtual Private Networks into Telecommunication Networks

618/1998 *Friedrich Eisenbrand and Andreas S. Schulz:* Bounds on the Chvátal Rank of Polytopes in the 0/1-Cube

617/1998 *Andreas S. Schulz and Robert Weismantel:* An Oracle-Polynomial Time Augmentation Algorithm for Integer Proramming

616/1998 *Alexander Bockmayr, Friedrich Eisenbrand, Mark Hartmann, and Andreas S. Schulz:* On the Chvátal Rank of Polytopes in the 0/1 Cube

615/1998 *Ekkehard Köhler and Matthias Kriesell:* Edge-Dominating Trails in AT-free Graphs

613/1998 *Frederik Stork:* A branch and bound algorithm for minimizing expected makespan in stochastic project networks with resource constraints

612/1998 *Rolf H. Möhring and Frederik Stork:* Linear preselective policies for stochastic project scheduling

611/1998 *Rolf H. Möhring and Markus W. Schäffter:* Scheduling series-parallel orders subject to 0/1-communication delays

609/1998 *Arfst Ludwig, Rolf H. Möhring, and Frederik Stork:* A computational study on bounding the makespan distribution in stochastic project networks

605/1998 *Friedrich Eisenbrand:* A Note on the Membership Problem for the Elementary Closure of a Polyhedron

596/1998 *Andreas Fest, Rolf H. Möhring, Frederik Stork, and Marc Uetz:* Resource Constrained Project Scheduling with Time Windows: A Branching Scheme Based on Dynamic Release Dates

595/1998 *Rolf H. Möhring Andreas S. Schulz, and Marc Uetz:* Approximation in Stochastic Scheduling: The Power of LP-based Priority Policies

591/1998 *Matthias Müller–Hannemann and Alexander Schwartz:* Implementing Weighted b-Matching Algorithms: Towards a Flexible Software Design

590/1998 *Stefan Felsner and Jens Gustedt and Michel Morvan:* Interval Reductions and Extensions of Orders: Bijections to Chains in Lattices

584/1998 *Alix Munier, Maurice Queyranne, and Andreas S. Schulz:* Approximation Bounds for a General Class of Precedence Constrained Parallel Machine Scheduling Problems

577/1998 *Martin Skutella:* Semidefinite Relaxations for Parallel Machine Scheduling

Reports may be requested from: Hannelore Vogt-Möller
Fachbereich Mathematik, MA 6–1
TU Berlin
Straße des 17. Juni 136
D-10623 Berlin – Germany

e-mail: moeller@math.TU-Berlin.DE

Reports are also available in various formats from

`http://www.math.tu-berlin.de/coga/publications/techreports/`

and via anonymous ftp as

`ftp://ftp.math.tu-berlin.de/pub/Preprints/combi/Report-`*number*`-`*year*`.ps`